UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/802,291	03/08/2001	Ashley Saulsbury	16747015310	6894

20350 7590 01/26/2005

TOWNSEND AND TOWNSEND AND CREW, LLP
TWO EMBARCADERO CENTER
EIGHTH FLOOR
SAN FRANCISCO, CA 94111-3834

EXAMINER	
LI, AIMEE J	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 01/26/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/802,291	SAULSBURY, ASHLEY	
	Examin r	Art Unit	
	Aimee J Li	2183	

-- The MAILING DATE of this communication appears on th cover sheet with the correspond nce address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 November 2004 and 20 December 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-5 and 8-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-5 and 8-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *04 November 2004*.
4) ☒ Interview Summary (PTO-413) Paper No(s)/Mail Date. *20 December 2004*.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-5, 8-20, and new claims 21-22 have been examined. Claims 1, 8, 12, 16, 17, and 18 have been amended as per Applicant's request. Claim 6 has been cancelled as per Applicant's request. New claims 21-22 have been added as per Applicant's request.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed on record in the file: RCE and IDS as filed 04 November 2004; Extension of time for two months as filed 04 November 2004; and Amendment as filed 20 December 2004.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 1, 3, 5-6, 8-9, 11-12, 14, 16-17 and 20-22 are rejected under 35 U.S.C. 102(b) as being anticipated by the *Alpha Architecture Handbook* (hereinafter "Alpha").

5. Regarding claim 1, Alpha has taught a processing core that executes a compare instruction (see p.4-113 Sec. 4.10.8, CMPTLE instruction), the processing core comprising:

a. A plurality of general-purpose registers comprising a first input operand register, a second input operand register and an output operand register (see p.3-2 Sec. 3.1.3),

b. A register file comprising the plurality of general-purpose registers (see p.3-2 Sec. 3.1.3),

c. Comparison logic coupled to the register file, wherein

i. The comparison logic tests for at least two of the following relationships with the compare instruction alone: less than, equal to, greater than or no valid relationship (see p.4-113 Sec. 4.10.8, CMPTLE instruction). Here, the CMPTLE instruction performs both the less than and equal comparisons using one instruction. Also, because less than or equal is a mutually exclusive operation with greater than, the CMPTLE instruction inherently determines if an operand is greater than a second operand by determining if its not less than or equal to the second operand.

ii. The comparison logic with the compare instruction along produces a value (see p.4-113 Sec. 4.10.8, CMPTLE instruction), and

iii. The value represents at least two of the following relationships: less than, equal to, greater than or no valid relationship (see p.4-113 Sec. 4.10.8, CMPTLE instruction);

d. Decode logic which selects the output operand register from the plurality of general-purpose registers (see p.4-113 Sec. 4.10.8). Here, because the IEEE Floating Compare instruction includes a target register (Fc) which stores the outputted results of a comparison, it is inherent that decoding logic be present in any processor which implements the Alpha instruction set architecture so that the specified target register can be correctly located and written into.

e. A store path between the comparison logic and the selected output operand register, wherein the value is stored in the selected output operand register (see

p.4-113 Sec. 4.10.8). Here, because the IEEE Floating Compare instruction includes a target register (Fc) which stores the outputted results of a comparison, it is inherent that a path exist between the logic which performs the comparison and the target register specified by the instruction so that the specified target register can be correctly located and written into.

6. Regarding claim 3, Alpha has taught the processing core that executes the compare instruction as set forth in claim 1, wherein said decode logic selects the first and second input operand registers from the plurality of general-purpose registers (see p.4-113 Sec. 4.10.8). Here, because the IEEE Floating Compare instruction includes two operand registers (Fa and Fb), it is inherent that decoding logic be present in any processor which implements the Alpha instruction set architecture so that the specified operand registers can be correctly located and read from.

7. Regarding claim 5, Alpha has taught the processing core that executes the compare instruction as set forth in claim 1, further comprising:

a. A first load path between the first input operand register and the comparison logic (see p.4-113 Sec. 4.10.8),

b. A second load path between the second input operand register and the comparison logic (see p.4-113 Sec. 4.10.8). Here, because the IEEE Floating Compare instruction includes two operand registers (Fa and Fb) which contain the operands for the comparison, it is inherent that a path exist between the logic which performs the comparison and each of the operand registers specified by the instruction so that the specified operands can be correctly located and read from, allowing the comparison to be made correctly.

8. Regarding claim 6, Alpha has taught the processing core that executes the compare instruction as set forth in claim 1, wherein the output operator register stores a value indicating a relationship between the first and second input operator registers which is at least one of greater than, less than, equal to and not a number (see p.4-113, Sec. 4.10.8, Description).

9. Regarding claim 8, Alpha has taught the processing core that executes the compare instruction as set forth in claim 6, wherein the value is an integer (see p.4-113, Sec. 4.10.8, Description). Here, when the comparison result is false, a value of zero is written into the target register, with zero being defined as an integer.

10. Regarding claim 9, Alpha has taught the processing core that executes the compare instruction as set forth in claim 1, wherein:

a. The first input operand register is a double precision floating point data type (see p.3-2 Sec. 3.1.3, p.4-62 Sec. 4.7, and p.4-113 Sec.4.10.8),

b. The second input operand register is a single precision floating point data type (see p.3-2 Sec. 3.1.3, p.4-62 Sec. 4.7, and p.4-113 Sec.4.10.8),

c. The output operand register is a double precision floating point data type (see p.3-2 Sec. 3.1.3, p.4-62 Sec. 4.7, and p.4-113 Sec.4.10.8).

11. Here, Alpha has taught the IEEE Floating Compare instruction using the T_Floating data format (see p.4-113 Sec.4.10.8), which is defined as the IEEE Double Precision floating point standard (see p.4-62 Sec 4.7). If one of the registers in the instruction, whether it is an input operand register or the output operand register, is single precision, the data is stored as a single precision value (see p.4-62 Sec. 4.7) but operated on correctly by instructions that use double precision values, such as the Floating Compare instruction (see p.3-2 Sec. 3.1.3).

12. Regarding claim 11, Alpha has taught the processing core that executes the compare instruction as set forth in claim 1, wherein the register file comprises special purpose registers which cannot store an output operand (see p.3-2 Sec. 3.1.3). Here, the registers F31 is a special purpose register in the floating-point register file which only supplies and stores the value of zero regardless of what is written to it.

13. Regarding claim 12, Alpha has taught a method for performing a compare operation, the method comprising the steps of:

a. Decoding a compare instruction (see p.4-113 Sec. 4.10.8). Here, because the IEEE Floating Compare instruction includes two operand registers (Fa and Fb) and a target register (Fc), it is inherent that decoding logic be present in any processor which implements the Alpha instruction set architecture so that the operand registers can be located and read from and the specified target register can be correctly located and written into.

b. Configuring first and second paths between a register file and comparison logic (see p.4-113 Sec. 4.10.8). Here, because the IEEE Floating Compare instruction includes two operand registers (Fa and Fb) which contain the operands for the comparison, it is inherent that a path exist between the logic which performs the comparison and each of the operand registers specified by the instruction so that the specified operands can be correctly located and read from, allowing the comparison to be made correctly.

c. Configuring a third path between the comparison logic and the register file (see p.4-113 Sec. 4.10.8). Here, because the IEEE Floating Compare instruction

includes a target register (Fc) which stores the outputted results of a comparison, it is inherent that a path exist between the logic which performs the comparison and the target register specified by the instruction so that the specified target register can be correctly located and written into.

d. Comparing a first input operand and a second input operand with the compare operation alone to produce a result which indicates at least two of the following mathematical relationships between the first input operand and the second input operand: less than, equal to, greater than or no valid relationship (see p.4-113 Sec. 4.10.8, CMPTLE instruction). Here, the CMPTLE instruction performs both the less than and equal comparisons using one instruction. Also, because less than or equal is a mutually exclusive operation with greater than, the CMPTLE instruction inherently determines if an operand is greater than a second operand by determining if its not less than or equal to the second operand.

e. Coupling an output operand to a general-purpose register in the register file (see p.4-113 Sec. 4.10.8). Here, because the IEEE Floating Compare instruction includes a target register (Fc) which stores the outputted results of a comparison, it is inherent that a path exist between the logic which performs the comparison and the target register specified by the instruction so that the specified target register can be correctly located and written into.

14. Regarding claim 14, Alpha has taught the method for performing the compare operation as set forth in claim 12, wherein the configuring steps each comprise a step of addressing a general-purpose register in the register file (see p.4-113 Sec. 4.10.8). Here, because the IEEE

Floating Compare instruction includes two operand registers (Fa and Fb) and a target register (Fc), it is inherent that decoding logic be present in any processor which implements the Alpha instruction set architecture so that the operand registers can be located and read from and the specified target register can be correctly located and written into. Furthermore, because a register can't be read from or written to without knowing its address, inherent to the locating of the operand registers contained within the IEEE Floating Compare instruction is the addressing of those registers within the register file so they can be read out and/or written to.

15. Regarding claim 16, Alpha has taught the method for performing the compare operation as set forth in claim 12, wherein the comparing step comprises a step of converting a data type of at least one of the first or second input operands (see p.4-62 Sec. 4.7). Here, it is taught that data conversion takes place between single- and double-precision floating-point instructions. Alpha has taught the IEEE Floating Compare instruction using the T_Floating data format (see p.4-113 Sec.4.10.8), which is defined as the IEEE Double Precision floating point standard (see p.4-62 Sec 4.7). If one of the registers in the instruction, whether it is an input operand register or the output operand register, is single precision, the data is stored as a single precision value (see p.4-62 Sec. 4.7) but operated on correctly by instructions that use double precision values, such as the Floating Compare instruction (see p.3-2 Sec. 3.1.3).

16. Regarding claim 17, Alpha has taught a method for executing a compare instruction in a processor, the method comprising steps of:

a. Issuing the compare operation. While not taught explicitly, it is inherent in the operation of a processor which implements an instruction set that in order to execute an instruction it must be issued to the execution unit(s).

b. Comparing a first input operand and a second input operand to determine at least two mathematical relationships between the first and second input operands, wherein the compare instruction alone causes the comparing step (see p.4-113 Sec. 4.10.8, CMPTLE instruction). Here, the CMPTLE instruction performs both the less than and equal comparisons using one instruction. Also, because less than or equal is a mutually exclusive operation with greater than, the CMPTLE instruction inherently determines if an operand is greater than a second operand by determining if its not less than or equal to the second operand.

c. Determining an output operand indicative of the mathematical relationships (see p.4-113 Sec. 4.10.8, Description),

d. Storing the output operand in a general-purpose register of a register file, wherein the output operand alone indicates the at least two mathematical relationships between the first and second input operands (see p.4-113 Sec. 4.10.8, Description).

17. Regarding claim 20, Alpha has taught the method for executing the compare instruction in the processor as set forth in claim 17, wherein the general-purpose register is used to store operators from other types of instructions (see p.3-2 Sec.3.1.3). Here, the floating-point registers are general-purpose registers that are used as sources and targets for floating-point instructions, which means that the registers are inherently able to store operators from other types of instructions than the IEEE Floating Compare instruction.

18. Referring to claims 21 and 22, Alpha has taught wherein the value in the operand register can be written to a location outside of the processing core (see p.4-95 Sec. 4.8.5, Description).

Claim Rejections - 35 USC § 103

19. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

20. Claims 2, 4, 10, 15 and 18-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over the Alpha Architecture Handbook (hereinafter "Alpha"), as applied to claim 1 above, and further in view of Colwell et al., U.S. Patent No. 4,833,599.

21. Regarding claim 2, Alpha has taught the processing core that executes the compare instruction as set forth in claim 1, but has not explicitly taught wherein a very long instruction word includes a plurality of compare instructions.

22. However, Colwell has taught a processor that executes multiple conditional branch instructions that reside in a VLIW instruction in parallel to improve execution speed and reduce the delay associated with branch mis-predictions (see Col.1 line 40 – Col.2 line 18 and Col.3 lines 3-34). One of ordinary skill in the art would have recognized that it is desirable and a goal of microprocessor design to improve the speed and throughput of a microprocessor. Therefore, one of ordinary skill in the art would have found it obvious to modify the processor of the Alpha to process a plurality of conditional branch instructions in a VLIW instruction in parallel to improve the execution speed of the processor.

23. Regarding claim 4, Alpha has taught the processing core that executes the compare instruction as set forth in claim 1, but has not explicitly taught wherein the processing core issues a plurality of compare instructions at one time.

24. However, Colwell has taught a processor that executes multiple conditional branch instructions that reside in a VLIW instruction in parallel to improve execution speed and reduce the delay associated with branch mis-predictions (see Col.1 line 40 – Col.2 line 18 and Col.3 lines 3-34). It is inherent to the parallel execution of multiple instructions then is the parallel issue of multiple instructions. One of ordinary skill in the art would have recognized that it is desirable and a goal of microprocessor design to improve the speed and throughput of a microprocessor, and one method of doing so is to increase the instruction-level parallelism by issuing and subsequently executing multiple instructions in parallel (see Col.1 lines 12-26). Therefore, one of ordinary skill in the art would have found it obvious to modify the processor of the Alpha to issue a plurality of compare instructions at one time so they can be executed in parallel and thus increase the throughput and execution speed of the processor.

25. Regarding claim 10, Alpha has taught the processing core that executes the compare instruction as set forth in claim 1, but has not explicitly taught wherein the processing core further comprises a plurality of processing paths that are coupled to the register file.

26. However, Colwell has taught a processor with multiple processor clusters, each containing multiple processing paths (integer and floating point processors) with each processing path connecting to a register file (see Col.5 lines 47-55), so that multiple instructions can be executed in parallel each cycle, providing an increase in processor performance (see Col.1 lines 7-26). One of ordinary skill in the art would have recognized that it is desirable and a goal of microprocessor design to improve the speed and throughput of a microprocessor. Therefore, one of ordinary skill in the art would have found it obvious to modify the processor of the Alpha to

process a plurality of conditional branch instructions in a VLIW instruction in parallel to improve the execution speed of the processor.

27. Regarding claim 15, Alpha has taught the method for performing the compare operation as set forth in claim 12, but has not explicitly taught wherein a compare operation is comprised within a very long instruction word.

28. However, Colwell has taught a processor that executes multiple conditional branch instructions that reside in a VLIW instruction in parallel to improve execution speed and reduce the delay associated with branch mis-predictions (see Col.1 line 40 – Col.2 line 18 and Col.3 lines 3-34). One of ordinary skill in the art would have recognized that it is desirable and a goal of microprocessor design to improve the speed and throughput of a microprocessor. Therefore, one of ordinary skill in the art would have found it obvious to modify the processor of the Alpha to process a plurality of conditional branch instructions in a VLIW instruction in parallel to improve the execution speed of the processor.

29. Regarding claim 18, Alpha in view of Colwell has taught the method for executing the compare instruction in the processor as set forth in claim 17, but has not explicitly taught wherein the comparing step comprises at least two of the following steps:

a. Determining if the first input operand is less than the second input operand (see "CMPTLE" instruction of p.4-113 Sec. 4.10.8),

b. Determining if the first input operand is greater than the second input operand (see "CMPTLE" instruction of p.4-113 Sec. 4.10.8). Here, because less than or equal is a mutually exclusive operation with greater than, the CMPTLE

instruction inherently determines if an operand is greater than a second operand by determining if its not less than or equal to the second operand.

c. Determining if the first input operand is equal to the second input operand (see "CMPTLE" instruction of p.4-113 Sec. 4.10.8),

d. Determining if there is no valid relationship between the first input operand the second input operand (see "CMPTUN" instruction of p.4-113 Sec. 4.10.8).

30. The *Alpha Architecture Handbook* has taught the above comparisons as parts of two different instructions (see p.4-113 Sec. 4.10.8), but has not explicitly taught a single instruction with the ability to perform the four comparisons simultaneously. However, the comparison logic required to perform the four comparisons as described inherently exists (see paragraph 19 above). One of ordinary skill in the art would have recognized that executing the comparisons of two instructions as a single instruction in a single clock cycle increases the speed and throughput of a microprocessor. Therefore, one of ordinary skill in the art would have found it obvious to perform the four comparisons simultaneously when processing one instruction instead of separately for each of four instructions in order to increase the speed and throughput of the processor.

31. Regarding claim 19, Alpha has taught the method for executing the compare instruction in the processor as set forth in claim 17, but has not explicitly taught wherein the compare instruction is a very long instruction word which comprises a plurality of compare instructions which are processed in parallel down separate processing paths.

32. However, Colwell has taught a processor that executes multiple conditional branch instructions that reside in a VLIW instruction in parallel to improve execution speed and reduce

the delay associated with branch mis-predictions (see Col.1 line 40 – Col.2 line 18 and Col.3 lines 3-34). One of ordinary skill in the art would have recognized that it is desirable and a goal of microprocessor design to improve the speed and throughput of a microprocessor. Therefore, one of ordinary skill in the art would have found it obvious to modify the processor of the Alpha to process a plurality of conditional branch instructions in a VLIW instruction in parallel to improve the execution speed of the processor.

33. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over the *Alpha Architecture Handbook* (hereinafter "Alpha") as applied to claim 12 above, and further in view of Patterson and Hennessy, *Computer Organization and Design* (hereinafter "Patterson").

34. Regarding claim 13, Alpha has taught the method for performing a compare operation as set forth in claim 12, but has not explicitly taught wherein the method further comprises a step of enabling the comparison logic in an arithmetic logic unit.

35. However, Patterson have taught that logical operations, such as comparisons, are conventionally performed within an arithmetic logic unit because they contain the necessary hardware building blocks to perform comparisons, such as AND gates for equality comparisons (see p.230-231). One of ordinary skill in the art would have recognized the desire of a microprocessor designer to reuse hardware in order to minimize the space taken up by dedicated hardware on the chip. Therefore, one of ordinary skill in the art would have found it obvious to modify the processor of the Alpha to include its comparison logic in an arithmetic logic unit, so as to adhere to convention and reuse hardware which already exists, thus minimizing the spaced needed for dedicated hardware on the chip.

Response to Arguments

36. Applicant's arguments filed on 22 July 2004 and 20 December 2004 have been fully considered but they are not persuasive. The amended claims have not overcome the prior art of record. The added claim language states that the comparison instruction alone produces a value, the resulting value represents at least two relationships, and the result is stored in an output register. The instruction, CMPTLE, is a single instruction that tests whether a value is less than or equal to another and output a result to a register. The result of the CMPTLE instruction is the result of the two comparison operations performed for the CMPTLE instruction. Also, as is shown by ActionScript Dictionary under the description of the less than or equal to operator and FunctionX Logical Operators under Less Than or Equal, when expression 1 is greater than expression2, i.e. when expression1 is not greater than or equal to expression2, the result is false. When expression1 is not greater than expression2, i.e. when expression 1 is greater than or equal to expression2, the result is true. When performing a less than or equal to comparison, it is known that when its condition is found true, the opposite is false and vice versa. Therefore, the result represents two comparison operations.

Conclusion

37. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Aimee J Li whose telephone number is (571) 272-4169. The examiner can normally be reached on M-T 7:30am-5:00pm.

38. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

39. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

AJL
Aimee J. Li
19 January 2005

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100